# Efficient Constraint Processing for Highly Personalized Location Based Services

Zhengdao Xu Hans-Arno Jacobsen

Department of Computer Science and
Department of Electrical and Computer Engineering,
University of Toronto,
zhengdao@cs | *jacobsen@eecg{.toronto.edu}*

## 1 Overview

Recently, with the advances in wireless communications and location positioning technology, the potential for tracking, correlating, and filtering information about moving entities (i.e., generally speaking moving objects, such as automobiles, people, packages etc.) has greatly increased. Potential applications range from location-aware, selective information dissemination, personalized route planning, goods tracking, surveillance of group formation to buddy tracking.

The implementation of efficient tracking, correlation, and filtering schemes supporting millions of mobile objects, poses significant challenges. The knowledge of spatial, temporal, and causal relationship between moving objects would allow the support of highly personalized and effective location-based service (LBS) for tracking, correlating, and processing object positions, profiles, and trends. For example, shoppers walking on the street may want the advertisements to be displayed on their PDAs only when they are near a specific store (location-aware m-commerce); a driver only wants traffic information effecting his immediate future surroundings (proximity-based alerts); in a buddy tracking application, two (or more) friends may wish to be notified if they are within a certain distance of each other or within a certain distance of a point of demarcation (location-constraint correlation).

All these applications can be broadly classified as location-aware information dissemination tasks, which, as has been shown, can be well supported by the publish/subscribe paradigm [2]. In a publish/subscribe systems, clients exchange information by publishing events and subscribing to events of interest. The central component of this architecture is the event broker, which maintains all the subscriptions and matches incoming events against all subscriptions. Upon a subscription match, a subscriber is notified.

The above location-aware applications are supported by two communication styles, orthogonal to the coordination mechanism used. These are pull-oriented and push-oriented styles. Pull-oriented LBS works in the request/response manner, which requires the clients to poll the available service from the server who respond by returning the result. In contrast, in the push-oriented style, service initiation (after an initial registration or sign-up phase) is carried out by the server (the service provider.)

The push-oriented style is better suited for LBS, since it puts less strain on the mobile devices and network resources (the network is subjected to less polling.) For example, in the buddy tracking application, two (or more) users may wish to be notified when they are within a certain distance of one another. In this case, each of them needs to subscribe to this event, expressed by a location constraint, stating that they wish to be notified if the distance between them is within a certain range. Once the constraint is in the system, the location information of the moving objects is checked against all location constraints stored in the system. We assume that location position information is available, such as provided by GPS, network-enhanced GPS, ground-based sensors, or other location positioning technology [12]. When the constraint is satisfied, the notification is sent to the corresponding user.

Due to the scalability of publish/subscribe in terms of number of supported clients and event processing speed [4], we believe that this is an extremely promising paradigm for enabling push-oriented LBS. However, we argue that to support location-awareness in such system additional needs must be addressed. Especially, the tracking of moving entities and the efficient correlation of their locations and interests. Typi-

**Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004**

cally in such systems, a vast number of the constraints are expected to be evaluated in a timely manner. The scalability and the efficiency of these algorithms becomes an issue.

L-ToPSS (Location-based Toronto Publish/Subscribe System) is our research prototype that provides LBS in a push-oriented style. In this demonstration, we will focus on demonstrating an algorithm that efficiently evaluates two types of location constraints:

1. $n$-body constraint: constraints of the form $|p_1, p_2 ... p_n| \leq d$ that designates whether n moving points $p_1, p_2 ... p_n$ are in a circle with the given diameter d. $p_i$ $(1 \leq i \leq n)$ is the identifier of the object $i$; $p_i$ is further translated into the coordinate of object $i$ in the location matching engine at time $t$. Value $d$ is called alerting distance. Buddy tracking is an example of the $n$-body constraint problem, with $n$ equals to two.

2. $n$-body (static) constraint: constraints of the form $|A, p_1, p_2 ... p_n| \leq d_A$, (where $A$ is the coordinate of some static point) that designates whether $n$ moving points $p_1, p_2 ... p_n$ are within a given range from the static point $A$. Here $d_A$ is the alerting distance. The location-aware mobile commerce mentioned earlier represents an example of the $n$-body (static) constraint problem with the advertising publisher (the store) as a static body.

More formally, the problem we are solving can be stated as follows: *Given a set of constraints $C = \{c_1, c_2 ... c_m\}$, which designates the desired location constraint relationships among a set of n possibly moving points $P_t = \{p_1, p_2 ... p_n\}$ in the space at time t, find all constraints $c_i$ in $C$ that are satisfied.*

In the next section we briefly introduce related work. In Section 3, we sketch our algorithm. Section 4 presents the L-ToPSS research prototype. In Section 5, we describe the software demonstration.

## 2 Related work

Many spatial indexing schemes, like Time-Oblivious indexing in [1], dynamic external memory data structures [7] and R-tree based indexing [9], have been proposed, which can be used to index mobile objects in space. However, they do not solve the above mentioned location constraint matching problem. In the buddy tracking system [10] that is studied by Arnon Amir et al. a centralized 2D quadtree-based algorithm is introduced. However it only solves the 2-body problem. And a quadtree data structure is not straight forward to extend to multi-dimensions, which may be useful for the constraint evaluation for the past and future (e.g, add time as one dimension). They also assume the same alerting distance for every pair of the objects, which is a limitation since different location constraints will have different alerting distances. Other related work includes spatial database of moving object and query of location dependent data [3, 5].

## 3 Sketch of the algorithm

Our algorithm for location constraint evaluation uses the Kd-tree indexing, which is usually used to solve the orthogonal range search problem with static objects. Our contribution is that we improved it to efficiently process the location constraints among mobile objects.

To index moving objects in space, the whole space is partitioned into small equally sized partitions. Those partitions are well managed in a Kd-tree data structure. And the whole space can be expressed by the union of the partitions represented by all the nodes in the same level of the tree. After the whole space is partitioned, each object (mobile or static) is associated with a certain leaf node of the tree according to its current position. The distance between the partitions serves as a rough measure of the bounds of the distance between objects, which is lying inside those partitions. This can be further exploited to evaluate the constraints.

As an example of partitioning the space, Figure 1(a) shows how this works in the 2D space, where a 4km×4km square is partitioned into 8 partition and a corresponding tree is also constructed (see Figure 1(b)).

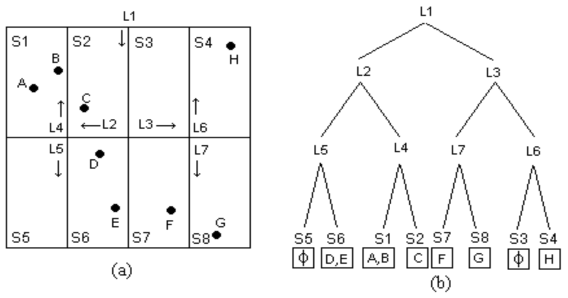

Figure 1: space partition: (a) 2D partition, (b) Kd-tree of the partition

We assume that our system is periodically informed of the position of the mobile objects by GPS or ground based sensors, for example. To keep track of the moving objects, a quick partition update management for those moving objects is very essential. In our system, when the new position of the mobile object is received, a backtracking partition update is performed, which backtracks from the leaf node of the tree up until the node representing the partition accommodates the current mobile object is found, and then the object is inserted from that node down to the appropriate leaf node.

When the constraint is evaluated, the partitions where objects in question are located is determined first; and based on the partitions they are in, we further make decisions for evaluation to reduce the need of distance computation. The technique we use to evaluate the $n$-body constraint is to use the partitions that are represented at the appropriate level of the tree, such that the distance between two objects is smaller than the alerting distance, if and only if, they are in the same or adjacent partitions. If all $n$ points are in the same or adjacent partitions, the smallest circle that encloses those $n$ points can be computed in $O(n)$ time [11]. For evaluating the $n$-body static constraint, we use the breadth-first search on the Kd-tree to decide the internal, bounding and external leaf partitions to the region with static object $A$ as the center and $d_A$ as the radius, depending on whether the partition is inside, intersecting with or outside the boundary of the region. Then the distance between the moving object and static object $A$ can be tracked according to the partition the moving object is in. The explicit distance computation is only needed for the moving object inside the bounding partition.

## 4 System architecture

L-ToPSS is our research prototype. The system architecture is shown in Figure 2. We assume that the matching of the publications against the subscriptions has been done by the filter engine and the identification numbers of the entities are sent to the location matching engine. The main component of our concern is the location matching engine, which stores the location constraints, as well as the link to the subscription and publication it is associated with. Our system

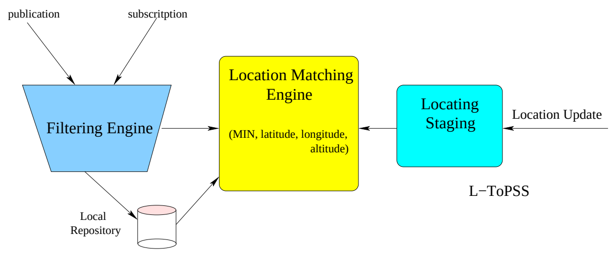

Figure 2: L-ToPSS System Architecture

model is very similar to that used in [2]. Each mobile publisher or subscriber is identified with a unique Mobile Identification Number ($MIN$). The subscription from some mobile user contains the $MIN$ of the mobile device on which the user wants to receive notifications. Similarly, the publication from some mobile publisher also contains a $MIN$. For a static publisher or subscriber, its location (latitude, longitude, altitude) is retrieved directly from the local database based on its identifier. The constraint is expressed in the form $(MIN_1, MIN_2, ...MIN_n, d)$ for the $n$-body constraint or $(id_{static}, MIN_1, MIN_2, ...MIN_n, d_{id_{static}})$ for the $n$-body (static) constraint; and they can be submitted into the system along with either the publications or the subscriptions.

The system receives the updates of mobile users' location and processes them in the location staging component. When the new update comes into the system, the corresponding location information and its timestamp are updated. The location information is represented as a ($MIN_i$, current_latitude, current_longitude, current_altitude) tuple. This tuple is forwarded to the location matching engine. All the constraints are managed in the location matching engine and they are evaluated periodically. During the evaluation, the system will first check the timestamps of all the mobile objects in the constraint to make sure that they are not obsolete. If the timestamps of all the objects are still valid, it matches them against the corresponding location constraint. We distinguish two types of constraints ($n$-body and $n$-body static) and the evaluation is based on the algorithm introduced the Section 3. Once the constraint is found satisfied, the subscribers associated with the constraint are notified with the publication they have subscribed. Through the demonstration, we show that by reducing the chance of distance computation, our algorithm speeds up the constraint evaluation thus allows the system to accommodate more constraints and serve more location-aware requests from the clients.

We are also aware of the system limitation mentioned in [2]; to avoid updating subscriptions with each location change of the subscriber, our location matching engine is processed independently from the matching between publications and subscriptions. The constraint is put into or retrieved from the system explicitly by the subscriber or the publisher when they subscribes or publishes.

In the next section, we present the demonstration of evaluation of constraints in the location matching Engine.

## 5 Software demonstration

In this demonstration, we will show the efficiency of our algorithm by evaluating a set of constraints for a given set of mobile objects in space. The constraints are submitted into or removed from the system in batch by the constraint generator. This can also be done for individual constraint through the demo panel.

The software setup for our demonstration is shown in the Figure 3. We use two IBM tools facilitating LBS evaluation: City Simulator [6], and Location Transponder [8]. The City Simulator generates dynamic spatial data simulating the motion of up to 1 million people and produces the trace-files that contain timestamped data records representing coordinate positions of the mobile objects. The Location Transponder takes the trace-files from the City Simulator as input and transmits the data to the location updates information to our system. We also develop a web application for insertion and retrieval of the constraints.

The evaluation is executed periodically, and once some constraint is satisfied, appropriate notification is sent out using different protocols.

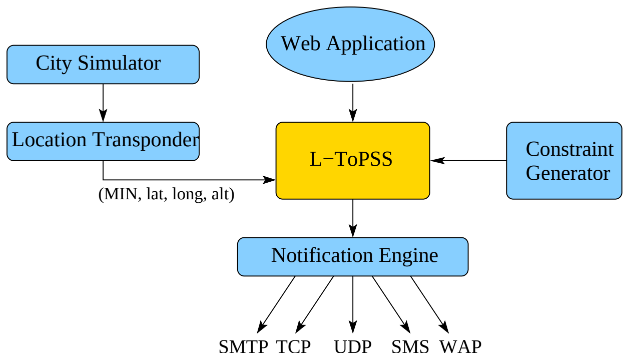

Figure 3: Demonstration Setup

To simulate the activity of a real-life system, our location matching engine stores millions of location constraints (both $n$-body and $n$-body static) that are to be evaluated against incoming location data of the mobile objects. To demonstrate the efficiency, we vary the number of objects in the constraints. Through the continuous evaluation, those satisfied constraints are alerted graphically to the users on the demo panel.

The performance of our algorithm is also compared with naive approach where the constraints are evaluated sequentially without the help of Kd-tree indexing. We will visualize the matching time for constraint evaluation, the number of explicit computations of the smallest enclosing disk and the number of partition update of the objects on the Kd-tree. Those statistics are shown on our demo panel when two methods are running as different processes on our pc. We will show through those statistics that even with little overhead for the partition update, our approach using Kd-tree indexing still outperform the naive approach.

We also show that by leveraging the size of the partition, the algorithm can reaches its best performance (least time needed for constraint evaluation and partition updating without sacrifices the accuracy).

## References

[1] Pankaj K. Agarwal, Lars Arge, and Jeff Erickson. Indexing moving points. In *Symposium on Principles of Database Systems*, pages 175–186, 2000.

[2] Ioana Burcea and Hans-Arno Jacobsen. L-topss - push-oriented location-based services. In *4th VLDB Workshop on Technologies for E-Services (TES'03)*, 2003.

[3] Susanne Hambrush Dmitri V. Kalashnikov, Sunil Prabhakar and Walid Aref. Efficient evaluation of continuous range queries on moving objects. *In DEXA 2002, Proc. of the 13th International Conference and Workshop on Database and Expert Systems Applications, Aix en Provence, France, September2-6 2002.*

[4] Françoise Fabret, H. Arno Jacobsen, François Llirbat, João Pereira, Kenneth A. Ross, and Dennis Shasha. Filtering algorithms and implementation for very fast publish/subscribe systems. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 30(2):115–126, 2001.

[5] Dimitris Papadis Yufei Tao Jun Zheng, Manli Zhu and Dik Lun Lee. Location-based spatial queries. *In SIGMOD Conference*, 2003.

[6] James Kaufman Jussi Myllymaki and Jared Jackson. City simulator. ibm alphaworks emerging technologies toolkit, http://www.alphaworks.ibm.com/tech/citysimulator, November 2001.

[7] George Kollios, Dimitrios Gunopulos, and Vassilis J. Tsotras. On indexing mobile objects. In *Proceedings of the Eighteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 31 - June 2, 1999, Philadelphia, Pennsylvania*, pages 261–272. ACM Press, 1999.

[8] Jussi Myllymaki and James Kaufman. Location transponder. ibm alphaworks emerging technologies toolkit, http://www.alphaworks.ibm.com/tech/transponder, April 2002.

[9] Simonas Saltenis, Christian S. Jensen, Scott T. Leutenegger, and Mario A. Lopez. Indexing the positions of continuously moving objects. In *SIGMOD Conference*, pages 331–342, 2000.

[10] Arnon Amir. Alon Efrat. Jussi Myllymaki. Lingeshwaran Palaniappan. Kevin Wampler. Buddy tracking - efficient proximity detection among mobile friends. In *Infocom 2004*.

[11] Emo Welzl. Smallest enclosing disks (balls and ellipsoids). In H. Maurer, editor, *New Results and New Trends in Computer Science*, LNCS. Springer, 1991.

[12] Y. Zhao. Standardization of mobile phone positioning for 3g systems. *IEEE Communication Magazine, July 2002.*